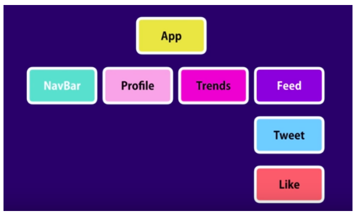
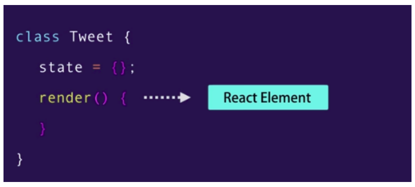
A REACT INTRODUCTION

--> Heart of react app : Components
--> Component : a piece of user interface
--> Building app with react : build a bunch of independent isolated and reusable Component and then compose them to build complex user interfaces
--> Every react app essensially a tree of components



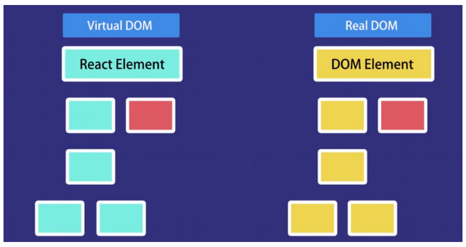
→ App has navbar, profile, tweets and feed
→ Feeds has Tweets
→ Tweets has Likes

--> In term of implementation, the component is typically implemented as js class that has some state and render method.



```
class Tweet {
  state = {};
  render() {  ·······▸ React Element
  }
}
```

// state : data that we want to display when when the component is rendered
// render method : responsible for describing what the UI should look like
--> Ouput of react render → react element



//React element : simple plain js object that maps to a DOM element.
--> It’s not a real DOM element it’s just a plain js object that represents that DOM element in memory.
--> React keeps a lightweight representation of the DOM in memory which we referred to as a virtual DOM. Unlike the browser or the real DOM, this virtual DOM is cheap to create
--> When we change the state of the component, we get a new react element, react will then compare this element and his children with the previous one, it figures out what is changed, and then it will update a part of the real DOM to keep it in sync with the virtual DOM

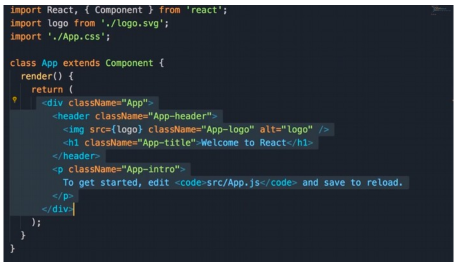
--> Which means in ract u no longer have to work with the DOM API in browser == we have no longer write code and query and manipulate the DOM or attach the event handlers to DOM element

B HOW TO INSTALL AND WHAT WE GET

--> Langkah2 (fully implementation):
1. install node js
2. npx create-react-app namanya
3. npm start
4. Third party: vs code with react snippet and prettier installed

--> What we get:
1. Development server
2. Webpack : for bunling our file
3. Babel : for compiling our js code
4. Other

--> What we get (in detail) :
1. 3 folder :
   a) /node module
   b) /public
      --> disini ada index.html yg ada <div id=”root”> //disinilah tepat kita meletakkan semua file react yang berasal dari /src
   c) /src
      --> berisi component2nya yang akan dimasukkan di <div id=”root”>
      --> defaultnya ada : app.css, app.js , dll
//app component



```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h1 className="App-title">Welcome to React</h1>
        </header>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}
```

// yang di block itu adalah JSX, kode JSX akan diconvert oleh babel ke plain js code yang browser dapat mengerti
//Syntax JSX mirip2 html dengan perbedaan :

C HELLO WORLD IN REACT
--> Langkah2 :
  a) in /src:
  1. buat index.js dan isi dengan :

```
import react from 'react';
```

//obj //module

```
import reactDOM from 'react-dom';

conts pertama =<h1> Hello word</h1>;
```

//ini adalah jsx yang merupakan react element dan merupakan bagian dari virtual DOM

```
ReactDOM.render(pertama, document.getElementById('root'));
```

//gunanya untuk merender jsx tadi ke real DOM
//jsxnya //tempat kita ingin meletakkan react elementnya ke real DOM
//root ada di <div> di /public/index.html
--> Contoh dengan dan tanpa babel:

```
<h1> Hello word</h1>;
```

--> Akan menjadi:

```
React.createElement( //ada 3 argument disini
"h1",                //type datanya
null,                //attribute
"hello word"         //isinya
);
```

D FIRST REACT COMPONENT
--> Langkah2:
  a. Install what u need:
    1) npx create-react-app counter-app
    2) npm i bootstrap@4.1.1
      --> di index.js:

```
import 'bootstrap/dist/css/bootstrap.css';
```

  b. Tambah Component Pertama:
    1) buat dir components di /src, buat file counter.jsx
    2) di counter.jsx isi dengan:

```
import React, {Component} from 'react';


 class Counter extends Component {
      render() {
        return <h1>Hello Word</h1>
```

//its jsx, aslinya dicompile oleh bible mjdi React.createElement (kaya diatas)

```
        }
 };
    export default Counter;
```

    3) di index.js tambahin dan ubah:
      //import default class Counter

```
import Counter from './components/counter';
```

      //ubah di DOM utama:

```
ReactDOM.render(<Counter />, document.getElementById('root'));
```

//yg akan dieksekusi(disini Counter class)

E EMBEDDING EXPRESSIONS 1
--> Jika element yang ada di jsx lebih dari 1, kita harus membungkusnya di <div> dan return diberi ()
  --> Karena argument pertama di React.Component adalah element, jika lebih dari 1 element , React.Element akan bingung
  --> exp: //antisipasinya adalah membungkus seluruh element dalam div

```
return(
   <div>
     <h1>Hello Word</h1>
     <button>Increment</button>
   </div>
   );
```

  --> Hasilnya:

```
React.createElement(
     "div",
     null,
     React.createElement(
       "h1",
       null,
       "Hello Word"
     ),
     React.createElement(
       "button",
        null,
       "Increment"
     )
 );
```
--> Cara lain jika nggk pengen nambahin <div>, pake <React.Fragment>:
```
  return(
    <React.Fragment>
      <h1>Hello Word</h1>
      <button>Increment</button>
    </React.Fragment>
    );
```

E EMBEDDING EXPRESSIONS 2

--> State is special property of react element basically its an object that includes any data that this component needs

--> jsx expressions are like normal js object, u can return them from a function, u can pass them to a function, u can also use them as value of a constant or variable
```
 class Counter extends Component {
     state = {
       count : 0
     };

     render() {
       return(
             <div>
                <span>{this.formatCount()}</span>
```
{/* in that curly brachets u can add any valid javascript expression */}
```
                  <button>Increment</button>
           </div>
          );
     }


     formatCount(){
```
/* cara cepat dari `this.state.count;` */
```
        const {count} = this.state;
         return count === 0 ? 'Zero' : count;
```
/* cara lain */
```
         return count === 0 ? <h1>Zero</h1> : count;
```
/* ini yng dimaksud : jsx expressions are like normal js object */
```
    }
 }
```

F SETTING ATTRIBUTE

--> Class in jsx is not class but className

--> Inline css in jsx: u must pass the value as an object, to do this the value must be in value of property

--> exp : di class Counter:
```
 state = {
   count : 0
 };
```
//inline css in jsx: property
```
 angka ={
   fontSize : 23,
```
//biasanya kan font-size, di jsx formatnya camelCase, dan px nggk perlu ditulis, cukup angkanya aja
fontWeight : "bold"
```
 }


 render() {
   return(
   <div>
     <span style={this.angka} className="badge badge-primary m-2">{this.formatCount()}</span>
```
{/*passing property angka ke inline css */}

{/*jika nggk pengen pake property, bisa langsung taro di dalem, tapi make nya double brachets{{ isi }} */}
```
   <span style={{fontSize:23}} className="badge badge-primary m-2">{this.formatCount()}</span>
     <button className="btn btn-secondary btn-sm">Increment</button>
   </div>
   );
 }
```

G RENDERING CLASSES DINAMICALLY

--> exp: if the value of property is zero, the padding color is yellow, when its not zero it will be blue: langkah2 :

1. add this function to change value of className's <span> based on value count:

```
ubahWarnaLencana(){
      let warna = "badge m-2 badge-";
      warna += this.state.count === 0 ? "warning" : "primary" ;
      return warna;
 }
```

2. di className <span> ubah valuenya menjadi function ubahWarnaLencana() :

```
<span className={this.ubahWarnaLencana()}>{this.formatCount()}</span>
```

H RENDERING LISTS

--> Nampilin semua yang di array menjadi <li>

--> in jsx there is NO LOOP, because jsx is not templating engine, its just simple syntax that eventually get compiled to React.Element

--> caranya: pakai array.map()

--> ract must identify each item in list, therefore we need a key for every item

--> Langkah2:

1) buat array dalam state:

```
state = {
  count : 1,
  tag: ['tag1','tag2','tag3']
};
```

2) di render() : buat <ul><li>, looping pakai array.map(), setiap <li> harus diberi key:setiap key harus unique:

```
<ul>
  {this.state.tag.map((a) => <li key={a}>{a}</li>)}
</ul>
```

I CONDITIONAL RENDERING

--> Kasus: Jika di array ada isinya nampilin gini, jika nggk ada akan nampilin gitu

--> DI jsx nggk ada loop, conditional statament, dll. Ngatasinnya pake:

1) Buat function di luar render dan panggil fungsinya di dalam render:

a) buat fungsi di luar render yang berisi if else condition:

```
renderTag(){
        if(this.state.tag.lenght === 0) return <p> There is no tag!</p>;
        return (
          <ul>
            {this.state.tag.map((a) => <li key={a}>{a}</li>)}
          </ul>
        );
 }
```

b) tampilin di render:

```
{this.renderTag()}
```

2) Pake combinasi boolean && string:

--> Beda dengan other programming language, u can apply the logical && between boolean value and string

--> exp :

1) `true && false` //false
2) `true && "hai"` //hai
3) `true && "hai" && "coba"` //coba --> diambil yang terakhir

--> implementasi: in render:

```
{this.state.tag.length === 0 && <p> There is no tag!</p>}
```

J HANDLING EVENTS

--> Contoh event : onClick(),onHover(), dll

--> Event di react:

1) Syntax :harus camelCase exp: onClick()

2) function di value event harus nggk pake () padahal dia fungsi, exp :

`naik() => {}` jika dipanggil ke dalam event menjadi : onCLick =(this.naik)

3) Ada hal istimewa di event,fungsi "this" pada event, reference ke window browser, karena dia stand-alone function klo di event

--> exp:

1) di render :

```
<button onClick={this.naik} className="btn btn-secondary btn-sm">Increment</button>
```

2) di event function :

```
naik(){
   this.state.count += 1; //hasil error property 'state' is undefined
}
```

--> Cara mengatasi ? ya di bind, cara ? go to next chapter

K BINDING EVENT HANDLERS

--> class di react, cuma boleh diisi, function, state, dan render, dan kita nggk bisa ngasih variable, jadi kita nggk bisa di luar function buat binding :

```
let a = this.state.count.bind(this);
```

--> Solusi:

1) Pake constructor buat dapetin "this" punya class :

```
constructor(){
   super();
   this.naik = this.naik.bind(this);
    //kenapa harus dimasukin di variable ? jawaban dibawah
   //this.naik = --> di ngeset fungsi naik yang ada di class Counter
   //this.naik.bind(this); --> nge bind 'this' punya naik agar selalu reference ke "this" punya Counter
}
```

2) Pake arrow function, tapi bakal deprecated di update selanjutnya:

```
naik = () => {
  console.log(this.state.count);  //hasil: sesuai nilai countnya
}
//lihat cara penulisan arrrow function, dia ada dalam variabel bernama naik
```

L UPDATING THE STATE

--> we cant update the value of property directly, exp:

--> di fungsi naik() kita kasih : this.state.count += 1; //ini nggk bakal work

--> di console terupdate, tapi react doesnt notice it, kenapa?

--> React pake virtual dom, jadi agar statenya terupdate kita pake method bawaan class Componet yaitu setState()

--> setState() : method ini memberi tahu react klo kita sedang mengupdate state, method lalu mensykronkan perubahan di virtual dom ke dom sebenarnya

--> exp:

```
naik = () => {
     this.setState({ count : this.state.count + 1})
 }
```

M WHAT HAPPENS WHEN STATE CHANGES

pass

N PASSING EVENT ARGUMENTS

--> kan method di eventHandler nggk ada tanda kurung (), trs gimana jika kita pengen ngasih argument?

--> ada 2 cara?

1) mbikin fungsi baru, buat ngisi argument, caranya:

a) buat fungsi dan fungsi yg berisi argument:

```
naik = (arg) => {
        //arg --> parameter
   console.log(arg);
   //apa yg dilakuin fungsi terhadap argument, disini cma console.log();
   this.setState({ count : this.state.count + 1})
 }

argNaik = () =>{
   this.naik("isi argument");
   //masukin argumentnya disini
 }
```

b) Panggil fungsi yang udah berisi argument:

```
<button onClick={this.argNaik} className="btn btn-secondary btn-sm">Increment</button>
```

2) Pake inline function:

--> mirip sama cara 1, tapi dijadiin inline aja fungsinya :

a) Mbikin fungsinya

```
naik = (arg) => {
         //kasih argument dan apa yg pengen dilakuin fungsi terhadap argument tsb
        console.log(arg);
        this.setState({ count : this.state.count + 1})
 }
```

b) Manggilnya

```
<button onClick={() =>this.naik('isi argument')} className="btn btn-secondary btn-
sm">Increment</button>
//wrapping fungsi naik() dalam onClick langsung
```